الحاسب الآلي

الصف الثالث الإعدادي

الترم الثاني

Chaptre on (Data)

data types:

On entering these data, it is temporarily stored in the computer memory (RAM), and to deal with these data we must assign a name and a type to it in memory, the type depending on the nature of the data element stored.

Visual Basic.Net Language provides many data types, which allow the user to store input values from the user, or values resulting from the execution of instructions and commands of the program.

Some Data Types provided by (Visual Basic.Net) Language are:

1- **Numeric Data Types:**
   a. **Integer Numeric Types**
      declared by: Byte- Short – Integer – Long .
   b. **Non-Integer Numeric Types** (decimal)
      declared by: Double- Single – Decimal .
2- **Character Data Types (declared by: String – Char).**
3- **Miscellaneous Data Types** (such as: Object – Date – Boolean)
   are those data that do not fall under the Numeric or the character types.

Notes :

- ✓ Each classification of Data Type has more than one type.
- ✓ Each data type **occupies a storage space in the memory**:
  for example, **the integer data type** occupies **4 bytes** of memory storage.
- ✓ Each data type has **a range of values (minimum value and maximum value)**:
  for example the range of values the data type (**Byte**) starts with (‘**0**’ and ends with ‘**255**’)

constants

They are places reserved in (RAM) that have data types.
on declaring them, name and data types are determined for them.

They take a fixed value and do not change during the progress of the program, such as: π, gravity acceleration, the speed of light and the speed of sound .etc.

Variables

They are places reserved in computer memory (RAM) that have data types.

declared and determined by its name and type (Data Type).
Its value usually changes during the running of the program.
The variable can take initial value, then value changes during the running of the program.

Such as: price of product - the value of the tax - the address of employee etc. .. All these data can be changed.

Naming rules of Constanta & Variables

1- Variable & constants names must begin with a letter or underscore (_)

2- Variable & constants names should not contain symbols or special characters (e.g.: ?,*,^, -, +,. etc.).

3- Do not use reserved words (Visual Basic.NET Language Keywords) such as (single, Dim, As).

Declaring constants

Use the (Const) in the declaration of the constants in the language of the VB.NET, as illustrated in the following syntax
Const Constant name As Data Type = Value

where:-

Const: decleration command about constants
Constant_Name: the name of the constant .
Data type : the type of data stored in the constant .
Value : the constant value that stored in the declared constant .

Example

Const C_Name **As** String = " جمهورية مصر العربية "
Const pi **As** Single = 22 / 7 OR **Const** pi **as** signle =3.14
Const BirthDate **As** Date = #1/25/2011#

Declaring Variables :

We use the Command (Dim) to declare a variable in VB.NET language, as shown in the following syntax:

Dim Variable_Name As Data Type [= Initial Value]

Where

Dim: is the declaration of variables. Variable Name: is a variable **name**. Data Type: is the data type to be stored in the variable.

Initial Value: is the initial value that is stored in the declared variable and this part is optional.

Example

```
Dim F_Name As String
Dim Total_Price As Single
Dim Today As Date = #1/25/2011#
```

Notes :

- ✓ The double quotes "" "" are used if the value of constant is a string value.
- ✓ The hashes # # are used if the value of constant is date or time.
- ✓ There are levels to declare the constants and variables you should take care of:

1- we declaration of the variable & constant in Event Procedure

2- we declare the variable (Radius) and constant (pi) on the level of class

Assignment statement

It is assigning a value to a constant or variable.

It is a statement that has two sides (right hand side and left hand side) separated by the **assignment operator** (=).

It takes the **value** on the **right side** of the assignment operator (=) and **stores it in the variable or constant** on the left.

Example: Area = 5 * 3

Left side	Assignment operator '='	Right side	example
Variable	=	Abstract value	A = 5
Variable	=	value of a variable	A = 5 B = A
Variable	=	value of Expression	A = 5 B = A + 3 * 2
Property	=	value of property	TextBox1.Text = "Egypt"

Notes :

- ✓ "**Me**" expresses the current window Form.
- ✓ Separates each variable and the other by the symbol **"&".**
- ✓ The reserved word (**vbCrLf**) is used to create a new line.
- ✓ Use the symbol (_) to write on more than one line if the code line is too long so you can organize and facilitate the process of reading the (Code).
- ✓ **Rem** word programmer can use it to the command (Rem) in writing remarks that can be referred to within the code, it is not compiled.
- ✓ **Run the program** by pressing the button (F5) to do (Start Debugging).

العمليات الحسابية :-

العملية	المعامل المؤثر	العملية	المعامل المؤثر
division	/	addition	+
exponent	^	subtraction	-
		multiplication	*

the priorities of performing calculations are:

1- Applying the process inside the brackets from the inside to the outside.
2- Applying the exponent.
3- Applying multiplication or division process from left to right, wherever comes first.

4- Finally, the Application of the addition or subtraction process from left to right, wherever comes first.
5- It is imperative to investigate the accuracy on writing mathematical expressions, so as to avoid falling into a (Logical Errors).

Errors : There are three types of errors

1- Syntax Errore

They are errors in the **common syntax commands of the language**.

Example:

Din x As Single

The variable (X) was declared but there is a mistake in writing the word (Dim)

Const x As Single

The constant (X) was declared but, its value is not assigned during the declaration.

This type of error is easy to detect because the (IDE) helps us as it does not allow any error of this type which displays the syntax of any command as you type it.

2- Logic Error

It happens when we get **incorrect results** after executing the program because of the **wrong formulating arithmetic or logic expressions.**

Example:

On calculating the area of a circle, we use the following code:
Dim Radius As Single Const x As Single = 22/7 Radius = TextBox1.Text Label2.Text = x + Radius ^ 2

- ✓ On this error it will not give any error messages, it will give the wrong result.
- ✓ To overcome this type of error, you must review the written code, and test the program with data already validated, to be sure there are no errors of this type (Logic Error).

3- Runtime Errors

These errors are discovered while running the program.

Example:

when declaring a variable of type Byte and during the program running, a value that is less than or greater than the allowable range is given, (less than (0) or greater than (255)) so an error appears during the run, meaning that the value is out of range.

Chapter Two
Branching

We need branching and executing a sequence of steps depending on the result of condition or the answer to a question.

You will find that writing the code of branching is just applying the algorithm with adherence to the general syntax rules used.

A) Branching using statement (If.... Then)

The general syntax of "If .. Then "

If Conditional Expression Then
Code
End If

The previous general syntax of "If .. Then " is conditional or branching statement.

- ✓ This means that if the conditional expression is true, the code will be carried out, then you will reach the end of the "If statement".

What is meant by "conditional expression"?

It consists of three parts :

1- logical operator preceded by an abstract value .
2- Value of a variable or constant or a result of a mathematical expression that is compared with an abstract value.
3- Value of a variable or constant or a result of a mathematical expression.

- ✓ If this conditional expression result is "**True**" the Specific code is executed.
- ✓ If the conditional expression result is "**False**" another code is executed.

logical operator

Relationship	operator	Relationship	operator
Greater than or equal	>=	Greater than	>
Smaller than Or equal	<=	less than	<
Not equal	< >	equal	=

The following table illustrates some examples:

Example of conditional expression	Conditional Expression		
	After logical operator	There are (6) logical expression	Before logical operator
If A > 5 If A < 5 If 5 < > A	Abstracted value	Greater than > less than < Smaller than Or equal to <= Greater than or equal > = equal = Not equal < >	Variable or constant
If B <= A If B >= A	Variable		
If B = A + 3 * 2 If C < > A – 3 * 2 If A^2 = B/C	a value from expression		

Example

```
If X >= 50 Then
MsgBox ("ناجح ")
End if
```

Exercise: Write program code which we enter student's score then the message "successful" appears in the message box if the score greater than or equal 50

```
Dim x as single
X=me.textbox1.text
If x >=50 then
    Msgbox("ناجح")
End if
```

Or This (If) statement can be written, in one line without writing (End if) as follows

```
Dim x as single
X=me.textbox1.text
If x >=50 then Msgbox("ناجح")
```

Notes :

When you enter any value less than 50, the MessageBox does not appear because the result of the condition is (False).
so the statement after (End if) which is (End Sub) will be executed.

B) Branching using statement (If.... Then .... Else)

This syntax is used if there is "Code1" that will be executed if the result of condition is "**true**", or another code "Code 2" is executed if the result of condition is "**False**".

If Conditional Expression Then

Code1 **(**The code in case of **True)**

Else

Code2 **(**The code in case of **False)**

End If

Exercise: Write program code which we enter student's score then the message "ناجح" appears in the message box if the score greater than or equal 50 Or the message "راسب" appears in the message box if the score Less than 50

```
Dim x as single
X=me.textbox1.text
If x >=50 then
    Msgbox("ناجح")
Else
    Msgbox("راسب")
End if
```

Or This (If) statement can be written, in one line without writing (End if) as follows

```
Dim x as single
X=me.textbox1.text
If x >=50 then Msgbox("ناجح") Else Msgbox("راسب")
```

Exercise: Write program code which we enter value stored in the variable (N) through a Textbox and displays th message "الرقم زوجى" or "الرقم فردى" in messagebox

```
Dim N as single
N=me.textbox1.text
If N mod 2 =0 then
    Msgbox("الرقم زوجى")
Else
    Msgbox("الرقم فردى")
End if
```

Notes :

- ✓ The conditional expression (N Mod 2) = Number divisible by 2 without a remainder.
- ✓ The function (Mod) returns the remainder of dividing the variable (N) by 2
- ✓ If the remainder of the division is equal to zero, this means the condition is (True), then a message "even number" appears in a message box.
- ✓ And if the remainder of the division is not equal to zero, this means that the condition is not met (False), and a message "odd number" appears in the message box.

C) Branching using statement (Select.... Case)

"**Select ... Case**" statement is used in Branching depends only on the **value of one variable** and **there are many conditions**, which reduces the code and makes it easier and clearer.

The syntax of (Select …Case) statement

```
Select …Case Variable

    Case value1
    code

    Case value2
    code

    Case value3
    code
    ….

    Case else
    code

End Select
```

Exercise: Write program code which entering temperature through the textbox and the phrase "above zero" or "zero" or "below button" is displayed through label1

```
Dim degree as single
Try
Degree =Me.textbox1.text
Select case degree
   Case 0
      Me.label1.text="صفر"
   Case < 0
      Me.label1.text="تحت الصفر"
   Case > 0
      Me.label1.text="فوق الصفر"
End select
Catch ex as exception
Mesgbox ("أدخل عدد")
Me.textbox1.focus( )
End try
Me.textbox1.text=""
```

Exercise: Write program code which show color name in textbox

```
Select case combobox1.selectedindex
   Case 0
      Textbox1.text="اللون هو اللون الأحمر"
   Case 1
      Textbox1.text="اللون هو اللون الأصفر"
   Case 2
      Textbox1.text="اللون هو اللون الأخضر"
   Case 3
      Textbox1.text="اللون هو اللون الأزرق"
   Case 4
      Textbox1.text="اللون هو اللون الأسود"
   Case 5
      Textbox1.text="اللون هو اللون الأبيض"
   Case Else
      Msgbox("يرجى اختيار احد الألوان من القائمة ")
End select
```

Chapter Three
Looping and Procedures

In this chapter, you will learn how to repeat a certain code for a number of times (which is called Loops) using (For… Next) and (Do While… loop).

1- Using (For... Next)

It is one of the limited loop statements used when we want to repeat a code for specific number of time.

General syntax for this statement

For Variable = Start Value **To** End Value **Step** Add Value
Code
Next [Variable]

Where:

1- Variable: is the name which represents the counter and its type must be numeric (integer or decimal).
2- Start Value: is the start value of the counter or the beginning of repetition is a numeric value.
3- End Value: is the value of the end of the counter and the end of the repetition is also a numeric value.
4- Add Value: is the increment value of the counter or value over the counter until it reaches the end value.
5- Code: is command or more to be replicated and be between the beginning of the loop (for) and its end (Next).

Notes:

- ✓ If the value of the increment is positive 1, it can be dispensed with writing Step Add Value.
- ✓ The default value to increase the counter is positive 1.

2. Typing a variable name counter next to "Next "optional.

Exercise :

1- Write program code to displayed number from 1 to 3 in messagebox on pressing button .

Dim M as integer For M=1 to 3 step 1 Msgbox(M) Next M	Dim M as integer For M=1 to 3 Msgbox(M) Next

Notes

- ✓ The end of the loop where the program returns to "For statement" testing the counter skip to the end value of the loop, if the counter value is less than or equal to the end value , the counter increases with the value of the increase and implements steps repetition
- ✓ Run the program by pressing (F5),

2- Modify the code to the printed numbers inside the TextBox as follows:

```
Dim m as integer
  For m=1 to 3 m
     Me.textbox1.text = Me.textbox1.text & m
  Next M
```

Notes :

- ✓ Using the concatenating operator "&" to concatenate the two strings together.
- ✓ vbCrLf is a A string constant used to add new line .
- ✓ vbCrLf = Visual Basic Carriage Return Line Feed

```
Dim m as integer
  Me.textbox1.text = " "
  For m=1 to 3 m
     Me.textbox1.text = Me.textbox1.text & m & vbCrLf
  Next M
```

Exercise :

1- Write program code to print multiplication table of 3 in textbox .

```
Dim m , product as integer
Dim str as string
Me.textbox1.text = " "
For m = 1 to 12
    Str = 3 * & " x " & m & " = "
    Product = 3 * m
    Me.textbox1.text = Me.textbox1.text = & str & product & vbCrLf
Next m
```

Exercise :

1- Write program code to print multiplication table for any number inserted into textbox .

```
Dim m , product , NUM as integer
Dim str as string
NUM = me.textbox2.text
Me.textbox1.text = " "
For m = 1 to 12
    Str = NUM * & " x " & m & " = "
    Product = NUM * m
    Me.textbox1.text = Me.textbox1.text = & str & product & vbCrLf
Next m
```

Example For ..... Next

Example	The code
To display the odd number from 1to 10	For i = 1 To 10 Step 2 Me.TextBox1.Text = TextBox1.Text & m & vbCrLf Next
To display the even numbers from 2 to 10	For i = 2 To 10 Step 2 Me.TextBox1.Text = TextBox1.Text & m & vbCrLf Next

Example	The code
To display the numbers that can be divided by 3 starting from 3 to 20	For i = 3 To 20 Step 3 Me.TextBox1.Text = TextBox1.Text & m & vbCrLf Next
Display numbers from 1.50 to 0.5 with decremented by 0.05 each time	For i = 1.50 To 0.5 Step -0.05 Me.TextBox1.Text = TextBox1.Text & m & vbCrLf Next
To display the numbers from 1 to the value of B at increasing value of C	For i = 1 To B Step C Me.TextBox1.Text = TextBox1.Text & m & vbCrLf Next
To display even numbers in descending order from 10 to 1.	For i = 10 To 1 Step -2 Me.TextBox1.Text = TextBox1.Text & m & vbCrLf Next

From the previous examples, we conclude that :

- ✓ We can determine the rate of increment of the variable after (Step) and then type a numeric value or numeric variable.
- ✓ The rate of increment should be **negative** if the starting value is greater than the end value .
- ✓ The starting value, the end value, or the increase rate can be a decimal number; in this case the loop variable type should be defined to accept decimals .
- ✓ The starting value, the end value, or the increase rate can be variable

2- Using (Do While ... Loop)

- ✓ **Do while ... loop** statement is used to repeat a specific code for a several times of an unknown end, based on a specific condition.
- ✓ **Do while … loop** are useful if you do not know the number of iterations emphatically.
- ✓ **Do while … loop** repeat a code until a condition comes true
- ✓ **Do while … loop** the code can be executed as long as the condition of the loop is true,

The general syntax of this statement is:

Do While Conditional Expression (شرطي تعبير)
Code
Loop

Notes :

- ✓ The code between the beginning of the loop "Do While" and its end will be implemented as long as the conditional expression is true.
- ✓ If the condition is not met for any reason, we get out of the iterative loop, and implement the code after the Loop if it exists.

Exercise :

1- Write program code to Enter a number in the Textbox print odd number or even number in listbox .

First method	Second method
Dim N, i As Integer N = TextBox1.Text ListBox1.Items.Clear() For i = 1 To N Step 2 ListBox1.Items.Add(i) Next	Dim N, i As Integer N = TextBox1.Text ListBox1.Items.Clear() i = 1 Do While i <= N ListBox1.Items.Add(i) i = i + 2 Loop

2- Write Program code that receives a positive number and displays the sum of odd or even numbers in the label (up to the number entered)

First method	Second method
Dim N, i, sum As Integer N = TextBox1.Text For i = 1 To N Step 2 sum = sum + i Next Label3.Text = sum	Dim N, i, sum As Integer N = TextBox1.Text i = 1 Do While i <= N sum = sum + i i = i + 2 Loop Label3.Text = sum

procedures

Procedure : a set of commands and instructions under a name, can be recalled by that name.

Notes :

- ✓ To implement them, and create a (Sub) if we have a set of commands that are frequently used in more than one place in the class.
- ✓ You can declare Procedures once and then, you recall the procedures many times from anywhere in your program.
- ✓ Use procedures avoids code duplication in places where procedures are recalled
- ✓ In the procedure declaration, we can use more than one Parameter.
- ✓ When the procedure is recalled, we determine values of the outside procedure called (Argument).
- ✓ when you add a new form window, a new class is created as Form1.
- ✓ Within the scope of this class we declare:
  1- Event procedures.
  2- Variables.
  3- Constants

The general syntax for declaring Sub is:

Sub Name (Parameters)
Code
End Sub

Where:

1- **Name** : reflects the name of the procedure.
2- **Parameters :** reflect the values that were used inside the procedure code that are used on recalling the procedure.
3- **Code :** is a set of orders and instructions carried out on recalling the procedure (Sub).

Exercise :

Declaring and calling a procedure :

```
Public Class Form1
    Dim total As Integer

    Private Sub Button1_Click(ByVal sender As System.Ob
        '10 عرض الأعداد الفردية من 1 إلى
        ShowOddOrEven()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Ob
        '10 عرض الأعداد الزوجية من 1 إلى
        ShowOddOrEven()
    End Sub

    Sub ShowOddOrEven()
        Dim i As Integer
        Label1.Text = ""
        For i = 1 To 10 Step 2
            Label1.Text = Label1.Text & " " & i
        Next
    End Sub

End Class
```

Recalling a (Sub) Procedure

Recalling a (Sub) Procedure

Declaring a (Sub) Procedure

Code executed when you recall the (Sub) procedure

Exercise :

Declaring and calling a procedure :

```
Sub ShowOddOrEven(ByVal Start As Integer)
    Dim i As Integer
    Label1.Text = ""
    For i = Start To 10 Step 2
        Label1.Text = Label1.Text & " " & i
    Next
End Sub

Private Sub Button1_Click(ByVal sender A
    '10 عرض الأعداد الفردية من 1 إلى
    ShowOddOrEven(1)
End Sub

Private Sub Button2_Click(ByVal sender A
    '10 عرض الأعداد الزوجية من 1 إلى
    ShowOddOrEven(2)
End Sub
```

Declaring a Parameter

Using this Parameter

Setting an Argument value

Function :

- Is a set of commands under a particular name that should express its task.
- It is applied to Parameters and returns a value.

The general syntax for declaring Sub is:

Function Function Name (**Parameters**) As **DataType**
Code
Return Value
End Function

Where:

1- **Name** : expresses the name of the function.
2- **Data type** : identifies the type of the returned value of the function.
3- **Parameters** : represents the parameters that will be used in the code.
4- **Code** : is a set of commands and instructions that will be executed on calling the Function.
5- **Value** " is the returned value by the function.

Exercise :

Declaring and calling a Function that calculate the sum of two numbers .

```
Public Class Form5
    Function Sum(ByVal First As Single, ByVal Second As Single) As Single
        Dim total As Single
        total = First + Second
        Return total
    End Function
End Class
```

```
 Private Sub Button1_Click(ByVal sender As
        Dim x As Single = TextBox1.Text
        Dim y As Single = TextBox2.Text
        Label4.Text = Sum(x, y)
 End Sub
```

Notes :

- ✓ We declared the (Sum) Function.
- ✓ Sum function Data Type (Single)
- ✓ Sum function receives two values (First and Second).
- ✓ Sum function return the value (total).

Notes:

- ✓ **Variables**:
  1- We can assign values for the Variables; during the declaration and the execution of the Program instructions.
  2- we can using the stored values.
- ✓ **Constants**:
  1- We should assign values to Constants; during the declaration only.
  2- we can using the stored values.
- ✓ **Functions**: It is recalled and returns a value in the light of the values assigned to the function.

From the previous example, it's clear that:

1. we can declare a Function.
2. we can determine its Parameters.
3. we can specify the Function type.
4. we can type the code within the scope of this Function.
5. we can return a value using the Return statement.